Weekly Report(February.25,2019-March.1,2019)

Anonymous Author(s)
Affiliation
Address
email

Abstract

During the winter holiday, I learned several optimization algorithms first, then I also learnt a lot about classifiers and finished assignment 1 of cs231n.

1 Learnt and Did

Mainly I learned about classifiers and optimization algorithms. And I finished assignment 1, which gives me a big challenge, since I'm not familiar with python.

1.1 Optimization Algorithms

1.1.1 Gradient Descent

Usually we use this way to find the maximum value or the minimum value. We shall make it when the function is convex function.

Main formula:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - h_\theta(x_i))^2$$

partial derivation:

$$\theta_j' = \theta_j + (y_j - h_\theta(x_i))x_j^i$$

To get the value we want, we need to update parameters in the oppsite direction of gradient.

1.1.2 Logistic Regression

It is usually used to dedal with two classification problems since its unique attributes.

Main formula:

$$Y_w = \frac{1}{1 + e^{-W^T X}}$$

Log likelihood function:

$$L(W) = -\frac{1}{m}\log(\prod_{i=1}^{m}[Y_w(X_i)]^{y_i}[1 - Y_w(X_i)]^{1-y_i})$$

Iterative formula(α is learning rate):

$$W_j = \frac{\alpha}{m}\sum_{i=1}^{m}(Y_w(xi) - y_i)x_{ij}$$

1.1.3 Newton's method

Newton's method drops faster than gradient descent. However, it works bad when dealing with hessian matrix.

Main formula:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f"(x_n)}$$

Hessian matrix:

$$x_{n+1} = x_n - \frac{J_f(x_n)}{H(x_n)}$$

$$J_f(x_n) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_n} \end{bmatrix}$$

$$H(x_n) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

1.2 Classifiers

1.2.1 KNN

KNN, K Nearest Neighbor, One of the most easiest classifiers.

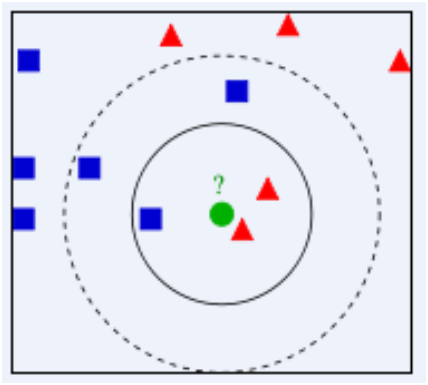

Figure 1: 2-dimension

In the training process of this algorithm, it stores all the input and output labels of training samples. in the test process, the distance between the test sample and each training sample L1 or L2 is calculated, then the first k training sample closest to test sample are selected. Then we'll vote on the label of the k samples, the category with the largest number of votes is categorized as test samples.

KNN is very easy, however it gets a bad result with classification accuracy of 27 percent.

1.2.2 SVM

SVM, Support Vector Machine, is a kind of generalized linear classifier which classifies data by supervised learning. Its decision boundary is the maximum margin hyperplane for solving learning samples.

Under normal conditions, we use linear-SVM to classify two kinds of data, between which there is an obvious interval. However, if it doesn't exist, we need to find it in higher dimension. The figures below show it.

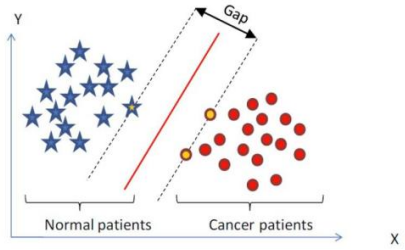

Figure 2: 2-dimension

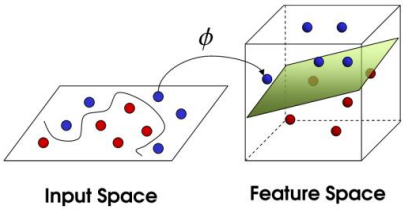

Figure 3: higher dimension

1.2.3 Softmax

In mathematics, the softmax function, also known as softargmax or normalized exponential function, is a function that takes as input a vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities.

That is, after applying softmax, each component will be in the interval(0,1), and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities. Softmax is often used in neural networks and Often acts as an activation function of a neural network

The standard (unit) softmax function $\sigma : R^K \to R^K$ is defined by the formula

$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for j = 1, ..., K and $\mathbf{z} = (z_1, \ldots, z_K) \in R^K$

1.2.4 Neural Network

Neural network is a very important part of machine learning. As I know, Convolutional neural network is the most powerful one. In assignment 1, I need to finish a two-layer neural network.

Formula:

Hyperplane: $y = WX + b$

Activation function: $f_{sigmod} = \frac{1}{1 + e^z}$

Activation function derivation: $f'_{sigmod} = f_{sigmod}(1 - f_{sigmod})$

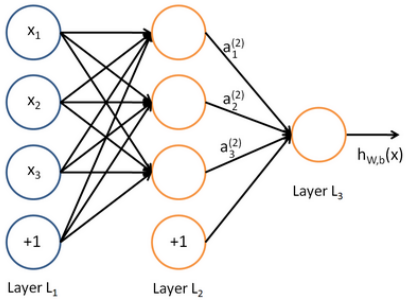

Figure 4: 2-dimension

2 Problems

The best problem I got is that I'm not familiar with Python libraries, such as numpy, matplotlib and so on. So was matrix operation. It wasted me a lot of time dealing with them and they still troubles me too.

3 Plan for Next Weeks

In the following weeks, I will keep going and finish assignment 2. I will still learn to use python and latex familiarly.